Assignment 3

Total = 30pts

AI Usage: You are able to use AI to help solve the problems as mentioned in class. However, your solutions should contain only PowerShell features that have been discussed in class. Answers that include features not discussed will lose points, even if the script works as expected. If you are unsure of what topics have been discussed, review the class notes or ask your instructor for clarification.

Create an array to hold your name and student number,
use array destructuring to extract the data into variables and then print out the values.

1 pt

In [147…

```
# add your code here
$studentInfo = @('Sam', 'McLaughlin', 'W0515523')
$firstName, $lastName, $studentId = $studentInfo
Write-Host "Name: $firstName $lastName StudentID: $studentId"
```

```
Name: Sam McLaughlin StudentID: W0515523
```

Name: Jane Doe, StudentID: w123456

Part A - Practice (15pts)

For each item below, determine the appropriate PowerShell code to generate the desired output.

---

1. Create an array of all layers in the OSI model.
   (e.g. "Physical", "Data Link", etc.)
   Then, loop through the array and print the names of all the layers,
   except the layer that begins with the letter 'T'.
   (Don't use that layer's full name when looking for it.)

   3 pts

In [9]:

```
# put your code here
$osiLayers = @('Physical', 'Data Link', 'Network', 'Transport', 'Session', 'Present
foreach ($layer in $osiLayers) {
    if ($layer[0] -ne 'T') {
        Write-Host $layer
    }
}
```

```
Physical
Data Link
Network
Session
Presentation
Application
```

2. Create an array of your courses this semester.
   Make each element another array, creating a multidimensional array.
   Have each row in the 2D array include both the course code and the course name.
   e.g. 'PROG1700' and 'Logic and Programming'
   Finally, print out the first course name in the list and the last course code.

   3 pts

In [10]:
```
# put your code here
# Creating a 2D array (jagged array) of courses
$courses = @(
    @('PROG1700', 'Logic and Programming'),
    @('COMP2000', 'Computer Systems'),
    @('MATH1500', 'Mathematics for Computing'),
    @('DBMS2200', 'Database Management Systems'),
    @('WEB2400', 'Web Development')
)
$firstCourseName = $courses[0][1]
$lastCourseCode = $courses[-1][0]

Write-Host "First course name: $firstCourseName"
Write-Host "Last course code: $lastCourseCode"
```

```
First course name: Logic and Programming
Last course code: WEB2400
```

3. Create a hash table that contains a list of common texting slang
   and their matching words/phrases (e.g. lol = laugh out loud)
   then use the new hash table to decode the following, unintelligle text.

   `idk imho fwiw ur skillz r l33t l8r`

   Display the decoded text on a single line.

   3 pts

In [30]:
```
# add your code here

$slangDictionary = @{
    'lol'  = 'laugh out loud'
    'idk'  = "I don't know"
    'imho' = 'in my humble opinion'
    'fwiw' = 'for what its worth'
    'ur'   = 'your'
    'skillz' = 'skills'
    'r'    = 'are'
    'l33t' = 'elite'
    'l8r'  = 'later'
}

$encodedText = 'idk imho fwiw ur skillz r l33t l8r'

$words = $encodedText.Split()

$decodedText = $words | ForEach-Object {
    if ($slangDictionary.ContainsKey($_)) {
        $slangDictionary[$_]
    } else {
        $_
    }
}

Write-Host ($decodedText -join ' ')
```

```
I don't know in my humble opinion for what its worth your skills are elite later
```

4. Create a program that takes a lowercase word,
   then makes every odd letter upper case and every even letter lower case.
   Print out the resulting word.

   e.g. `elite => ElItE`

   3 pts

In [101…
```
# use this word
$word = "hacker"

# add your code here
$result = ""
for ($i = 0; $i -lt $word.Length; $i++) {
    $char = [string]$word[$i]
    if ($i % 2 -eq 0) {
        $result += $char.ToUpper()
    } else {
        $result += $char.ToLower()
    }
}
Write-Host $result
```

```
HaCkEr
```

5. Create a program that produces an acronym from a series of words.
   e.g. if the input is `"Nova Scotia Community College"`, the output should be `"NSCC"`

   3 pts

In [89]:
```
$text = "Nova Scotia Community College"
$acronym = ""

$words_array = $text.Split(" ")
foreach ($word in $words_array) {
    $acronym += $word.Substring(0, 1).ToUpper()
}

Write-Host $acronym
```

```
NSCC
```

Part B - Bug Hunt (10pts)

For the following questions, locate the bug(s) hidden in the code.
Write a brief sentence describing the bug(s) in the space provided.
Then, fix the bug(s) in the code to verify your fix works.

Note: Just find and fix the bug(s), don't rewrite the script.

---

6. The following data was extracted from a log file, however,
   the data isn't in the desired format used in a weekly report.

   The script should take the dates in the format `"yyyy-mm-dd HH:MM:SS"` and convert them into the format `"mmm dd, yyyy"`, ignoring the time.

   e.g. `2024-06-14 12:53:05 AM` => `Jun 14, 2024`

   3 pts

Question: Describe what the bug is below.

Answer: Convert $mmtoanInteger : [int]$mm changes $mmfromastringtoaninteger. AdjustArrayIndex : Bysubtracting1, weensurethat "$ 06 months.Split(","), which is "Jun".

In [53]:
```
# convert this log entry
$logEntry = "2024-06-14 12:53:05 AM`t10000`tInformation`tStarting session 0"

#find the bug here somewhere
$months = "Jan,Feb,Mar,Apr,May,Jun,Jul,Aug,Sep,Oct,Nov,Dec"
$created, $id, $level, $message = $logEntry.Split("`t")
$date, $time = $created.Split()
$yyyy, $mm, $dd = $date.Split("-")
$mmm = $months.Split(",")[([int]$mm - 1)]

"Date`t`tID`tLevel`t`tMessage"
"$mmm $dd, $yyyy`t$id`t$level`t$message"
```

```
Date            ID      Level           Message
Jul 14, 2024    10000   Information     Starting session 0
```

7. The following script takes a hash table of all the assignments, quizzes and exams from this course. It should determine the final grade as a percentage and shows whether the student has passed or failed the course.

   4 pts

Question: Describe what the bugs are below.

Answer: There is a couple errors, the name typo "Assignments" vs "Assignment". incorrect quiz weight, 0.025 instead of 0.25.

In [63]:
```
$grades = @{
    'Assignments'= @(45, 65, 12, 78, 52, 87);
    'Quizzes'= @(89, 45, 67, 90);
    'Midterm'= 72;
    'Final'= 85
}
$assignmentGrade = 0
foreach ($grade in $grades['Assignments']) {
    $assignmentGrade += $grade
}
$assignmentGrade /= $grades['Assignments'].Count
$quizGrade = 0
foreach ($grade in $grades['Quizzes']) {
    $quizGrade += $grade
}
$quizGrade /= $grades['Quizzes'].Count
[int]$finalGrade = ($assignmentGrade * 0.45) + ($quizGrade * 0.25) +
                   ($grades['Midterm'] * 0.15) + ($grades['Final'] * 0.15)
```

```
Write-Host ("Final Grade: {0:N0}%" -f $finalGrade) -NoNewline
if ($finalGrade -ge 60) {
    Write-Host " (PASS)" -ForegroundColor Green
} else {
    Write-Host " (FAIL)" -ForegroundColor Red
}
```

```
Final Grade: 67% (PASS)
```

8. A simple method of encryption/decryption is known as **"ROT-13"**, a type of "substitution cipher", where letters in a message are replaced by other letters in the alphabet that are 13 away from the original letter.

   In effect the original alphabet is converted to the following:
   `ABCDEFGHIJKLMNOPQRSTUVWXYZ` => `NOPQRSTUVWXYZABCDEFGHIJKLM`

   The script below should take a given message and encrypt it using the ROT-13 cipher.
   e.g. `"nscc"` => `"afpp"`

   3 pts

Question: Describe what the bug is below.

Answer: $letter + 12 was the reason that the cypher was not working properly, it was only shifting the letters 12 down the alphabet. upon adjusting this to 13 the script works as intended

In [59]:

```
# message to encrypt
$message = "nscc"

$key = [ordered]@{}
for ($letter = 0; $letter -le 25; $letter++) {
    $key.Add([char]($letter + 97), [char](($letter + 13) % 26 + 97))
}
$encrypted = ""
for ($letter = 0; $letter -lt $message.Length; $letter++) {
    $encrypted += $key[$message[$letter]]
}
Write-Host "$message => $encrypted"
```

```
nscc => afpp
```

Part C - Practical (5pts)

The following is a list of new employees that have been hired at your company.

Ella Forester

Liam Jackson

Wendy Johnston

Noah Jenkins

Sophia Jordan

William Johnson

Ava Adams

Ethan Forestall

Isabella Ingram

Mason Mitchell

The IT department has been tasked with setting up new user accounts for each employee. In order to do this, they need to create a script that will generate a username for each employee based on the following rules:

- The username should be the first letter of the employee's first name, followed by the first 4 letters of their last name.
- The username should be all lowercase.
- If the username is already taken, add a number to the end of the username to make it unique.

Create a script that will take an array of the given employee names and generate a username for each employee based on the rules above. The script should output all the employees' names with their new username in the format " => "

e.g.

John Smith => jsmit Sally Jones => sjone Jack Smits => jsmit2 ...

In [90]:
```
# List of new employees
$employees = @(
    "Ella Forester",
    "Liam Jackson",
    "Wendy Johnston",
    "Noah Jenkins",
    "Sophia Jordan",
    "William Johnson",
    "Ava Adams",
    "Ethan Forestall",
    "Isabella Ingram",
    "Mason Mitchell"
)
$usernames = @{}

foreach ($employee in $employees) {
    $firstName, $lastName = $employee.Split(" ")
    $baseUsername = ($firstName[0] + $lastName.Substring(0, [math]::Min(4, $lastNam
    $username = $baseUsername
    $counter = 1
    while ($usernames.ContainsKey($username)) {
        $username = "$baseUsername$counter"
        $counter++
    }
    $usernames[$username] = $true
    Write-Host "$employee => $username"
}
```

```
Ella Forester => efore
Liam Jackson => ljack
Wendy Johnston => wjohn
Noah Jenkins => njenk
Sophia Jordan => sjord
William Johnson => wjohn1
Ava Adams => aadam
Ethan Forestall => efore1
Isabella Ingram => iingr
Mason Mitchell => mmitc
```